

Ingeniería del Software II

Parcial #2 – Análisis Estático de Programas

Ej1	Ej2	Ej3	Ej4	Nota
✓	✓	✓	✓	

El exámen es a **libro abierto**. Cada ejercicio se evaluará como **Bien**, **Regular** o **Mal**. Para aprobar el examen es necesario tener al menos 2 ejercicios Bien y al menos 1 ejercicio Regular.

Bien = 2,5p, Regular = 1,25p, Mal = 0p.

Ejercicio 1

Definir un análisis de data-flow que infiera, por cada punto del programa, si las variables hasta ese punto del programa contienen un valor cuya secuencia de cambios lo mantuvo siempre constante, monótonico no decreciente ($x_{i+1} \geq x_i$) o monótonico no creciente ($x_{i+1} \leq x_i$).

- Definir el reticulado de este análisis
- Indicar cuál es la semántica abstracta para la operación “+” de manera que modele de la forma más precisa posible la suma de dos naturales (i.e., $a, b \in \mathbb{N}_0$) para dicho reticulado.

a	b	a + b
...

Ejercicio 2

Dado el siguiente programa y el análisis del ejercicio 1:

```
def foo() -> nat:
  x:nat = 0
  y:nat = 5
  if y > 0:
    x ← x + 1
  else:
    x ← x - 1
  return x
```

Se pide:

- Exhibir las funciones de transferencia para la asignación y para los condicionales
- Construir el CFG del programa.
- Calcular el valor abstracto de las variables para los conjuntos IN y OUT de cada nodo en el CFG.

Ejercicio 3

Dado el siguiente programa en lenguaje IMP calcular el *points-to graph* con abstracción a nivel de allocation-site usando un algoritmo no sensitivo a flujo:

```
int main(int choice) {
  var a,b,c;
  a = 4;
  b = alloc 4;
  *b = 10;
  while(choice < 5) {
    c = alloc 4;
    b = &a;
```

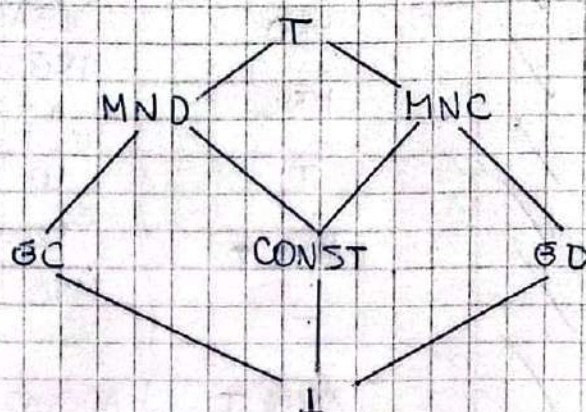
```
        if (choice == 1) {  
            c = b;  
        } else {  
            c = &a;  
        }  
    }  
}
```

Ejercicio 4

Deducir, paso por paso, la Weakest Precondition del siguiente programa:

```
void main (nat x) {  
    var z;  
    if (x != 0) {  
        z = x;  
    } else {  
        z = x+1;  
    }  
    { z > 0 }
```


① a) PROPONGO EL SIGUIENTE RETICULADO:



(DESPUES DE TERMINAR
1 Y 2 ME DI CUENTA
QUE R PODIA SER
MAS SIMPLE SIN
USAR EC Y ED. EL
MO NO ESTA MAL, SOLO
QUE ES MAS ESPECIFICO
QUE LO NECESARIO)

DONDE C ELEMENTO DE RETICULADO "REPRESENTA" LO SIGUIENTE:

T: TOP

MND: MONOTÓNICO NO DECRECIENTE ($x_{i+1} \geq x_i$)

MNC: MONOTÓNICO NO CRECIENTE ($x_{i+1} \leq x_i$)

EC: ECTRICTAMENTE CRECIENTE ($x_{i+1} > x_i$)

ED: ECTRICTAMENTE DECRECIENTE ($x_{i+1} < x_i$)

CONST: CONSTANTE ($x_{i+1} = x_i$)

1: BOTTOM

b)

a	b	$a \leq a+b$
EC	CONST	MND ✓
EC	EC	EC ✓
EC	ED	MND ✓
EC	MND	MND ✓
EC	MNC	MND ✓
MND	CONST	MND ✓
MND	EC	MND ✓
MND	ED	MND ✓
MND	MND	MND ✓
MND	MNC	MND ✓
CONST	CONST	CONST MND ✓
CONST	EC	MND ✓

(ESTO FUE ACLARADO QUE
DEBIAMOS CAMBIARLO)

NOTA

a	b		a = a + b	ABRIGO FILAS QUE ME FALTARON A CONTINUACIÓN		
CONST	ED	✓	MND	a	b	a + b
CONST	MND	✓	MND	EC	T	MND
CONST	MNC	✓	MND	MND	T	MND
MNC	CONST	✓	T	CONST	T	MND
MNC	EC	✓	T	ED	T	T ✓
MNC	ED	✓	MND T	MNC	T	T
MNC	MND	✓	T	ADemás, SIEMPRE QUE a SEA T, a+b SERÁ T		
MNC	MNC	✓	MND T			
ED	CONST	✓	MND T	SIEMPRE QUE a SEA L a+b SERÁ L		
ED	EC	✓	T			
ED	ED	✓	MND T	SIEMPRE QUE b SEA L a+b SERÁ L		
ED	MND	✓	T			
ED	MNC	✓	MND T			

④ $wp(\{x \neq 0\} \{z = x\} \text{ ELSE } \{z = x + 1\}, z > 0) =$

~~wp(x > 0, z > 0)~~

$$x \neq 0 \Rightarrow wp(z = x, z > 0) \wedge$$

$$x = 0 \Rightarrow wp(z = x + 1, z > 0) =$$

$$(x \neq 0 \Rightarrow x > 0) \wedge \underbrace{(x = 0 \Rightarrow \underbrace{x + 1 > 0}_T)}_T =$$

$$(x \neq 0 \Rightarrow x > 0) = \boxed{x = 0 \vee x > 0} \quad \checkmark$$

↑
LA WP RESULTANTE.

- ② a) ANTES DE DEFINIRLAS ACARO QUE EL ANALISIS SERA MAY FORWARD. POR ESA RAZON, LAS FUNCIONES DE TRANSFERENCIA LAS DEFINO PARA OUT.

ASIGNACION: PARA ESTO HICE UNA CONSULTA Y ME RESPONDIERON QUE SE REDUCE A:

① $X = m$ ~~XXXXXXXXXX~~

② $X = X + m$ ~~XXXXXXXXXX~~

③ $X = X - m$ ~~XXXXXXXXXX~~

DEFINO UNA FUNCION DE TRANSFERENCIA PARA C/CASO.

① $OUT[X] = \begin{cases} CONST & INC[X] = 1 \\ T & OTRO CASO \end{cases}$

ESTO ES ASI YA QUE AL NO SABER EL VALOR DE ~~X ANTES DE LA ASIGNACION~~ ANTES DE LA ASIGNACION, ENTONCES NO PODEMOS DECIR NADA SOBRE LA SUCCESION.

② $OUT[X] = \begin{cases} MND & INC[X] = MND \vee \\ & INC[X] = EC \vee \\ & INC[X] = CONST. \\ T & OTRO CASO \end{cases}$

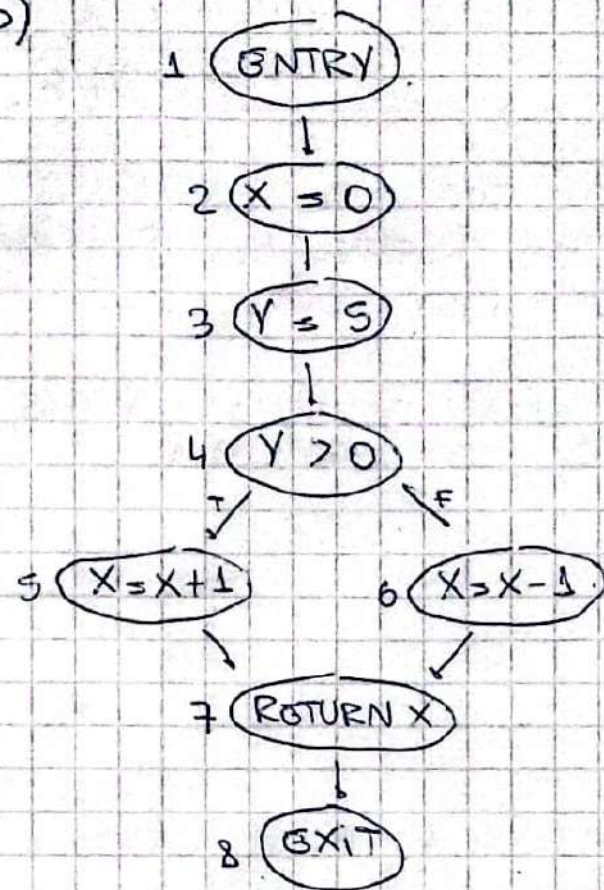
ACI, SI SABEMOS QUE X ES EST. O CRECIENTE O CONST, ENTONCES SABEMOS QUE SERA CRECIENTE EN LOS N DESPUES DE LA SUMA.

③ $OUT[X] = \begin{cases} MNC & INC[X] = MNC \vee \\ & INC[X] = EC \vee \\ & INC[X] = CONST. \\ T & OTRO CASO \end{cases}$

CONDICIONAL: EN LOS CONDICIONALES NO HAY CAMBIO EN EL VALOR DE LAS VARIABLES.

$OUT[X] = IN[X]$

b)



c)

m	IN[X]	IN[Y]	OUT[X]	OUT[Y]
1	-	-	↓	↓
2	↓	↓	CONST	↓ /
3	CONST	↓	CONST	CONST /
4	CONST	CONST	CONST	CONST /
5	CONST	CONST	MIND	CONST
6	CONST	CONST	MINC	CONST /
7	T	CONST	T	CONST /
8	T	CONST	T	CONST /

③

